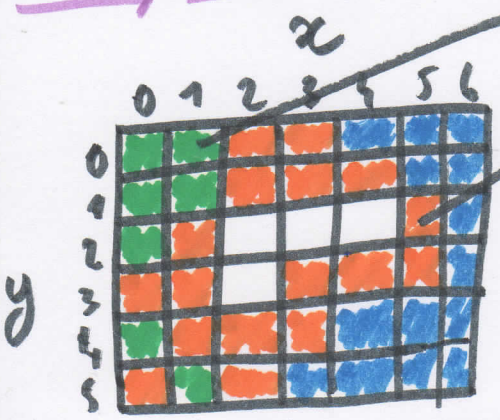
① Computer science 101 - Stanford

Digital images

For the computer, an image is a structure made of many little numbers, square "pixels", each showing one color.

- Each **pixel** is a small square that shows a single color
- An 800 x 600 image is 800 pixels wide by 600 pixels high, 480 000 pixels in all (0.5 megapixels)
- Digital cameras today produce images with several megapixels per image, say 8000 x 6000 for about 5 megapixels.

Image diagram



Pixel at x y (1,0)
red: 6 green: 250 blue: 7 (i.e shade of green)

Pixel at x y (5,2)

...

- The pixels are organized as a **grid**, identifying each by an (x, y) coordinate, so every pixel can be identified by its (x, y) address. E.g. x=0 y=0 top left pixel of the whole image.

RGB color scheme - Red Green Blue

The RGB scheme is one popular way of representing a color in the computer. Every color is defined as a particular combination of pure red, green, and blue light. It is different from mixing paints, here mixing light (see Newton).

② Computer science 101 - Stanford

RGB - Three numbers

Any color can be encoded as three numbers. One each for red, green and blue. A color is defined as a mixture of pure red green and blue lights of various strengths. Each of the red, green and blue light levels is encoded as a number in the range 0:255 with 0 meaning zero light and 255 meaning maximum light.

Color	Red number	Green number	Blue number
Red	255	0	0
Purple	255	0	255
Dark yellow	100	100	0
White	255	255	255
Black	0	0	0

Digital atomization

We can start with a whole, textured digital image of something. Then break it down into small square pixels. Then each pixel breaks down to 3 numbers in the range 0:255. This is a typical computer pattern.

Pixel $(\underbrace{1,0}_{x\ y})$ red: 6 green: 250 blue: 7

③ Computer science 101 - Stanford

Computer hardware

- "Hardware" refers to the physical parts of the computer, and "software" refers to the code that runs on the computer. In music for example, the piano would be the hardware and the music the software

Chips and Transistors

- Modern computers use tiny electronic components which can be etched onto the surface of a silicon chip (⚠ ≠ from silicone, the soft rubbery material). The most common electronic component is the 'transistor' (invented early 1950's). It is a basic building block used to construct more complex electronic components. chips = puce (FR)

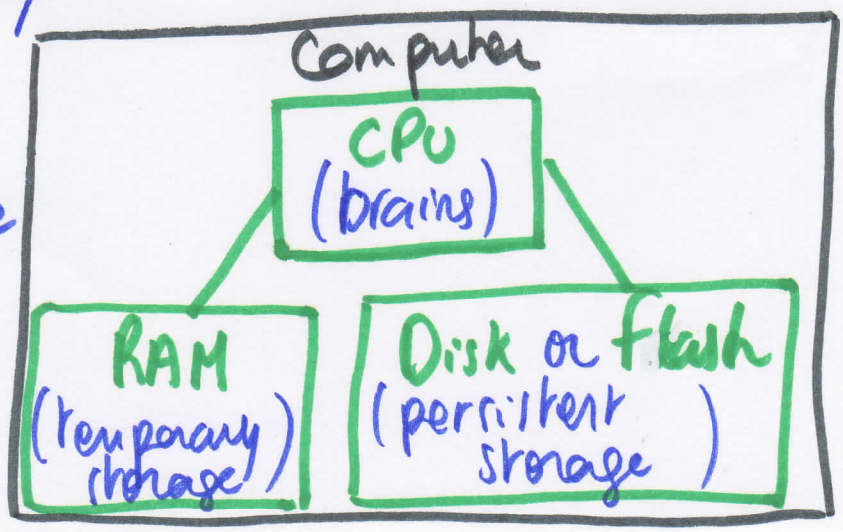
E.g. CPU chips, memory chips, flash chips, ...

Moore's law (Intel co-founder) → more like an observation

- Density of transistors on a chip doubles about every 18-24 months
- Per dollar, computer technology gets exponentially better as time goes along and keeping capability constant, computers get cheaper and cheaper.

Computer Hardware

Three major parts that make up a computer. They are found in a laptop, smart phone or tablet



(4) Computer science 101 - Stanford

CPU - Central Processing Unit

It refers to the "brains" of the computer. It does the **active "running"** of code, manipulating data, while the other components have a more passive role, such as storing data. We say that a computer can run 2 billion operations per second, that's the CPU.

Bytes

A byte is a **unit of information storage** (a document, an image, a movie ... how many bytes?). 1 byte is enough to hold 1 letter, like "a". Storages in RAM, hard drives and flash drives are all **measured in bytes**, despite being very different hardwares.

- Megabyte, MB, about 1 million bytes
- Gigabyte, GB, about 1 billion bytes

RAM - Memory : Random Access Memory

RAM is the working scratchpad memory the computer uses to store code and data that are being **actively used**. It is a storage area of bytes under the control of the CPU. RAM is relatively fast, able to retrieve the value of any particular byte in a few nanoseconds (1 nanosecond = 1 billionth of a second). The other main feature of RAM is that it only keeps its state so long as it is supplied with power - gone when power goes out. It is "volatile" not persistent

⑤ Computer science 101 - Stanford

Persistent storage: Hard drive, flash drive

- Persistent storage: long term storage for bytes as files and folders, even when power is removed. The **hard drive** reads and writes magnetic patterns on a spinning metal disk to store the bytes, while flash is "solid state" (also known as SSD - Solid State Disk), no moving parts, just silicon chips with tiny groups of electrons to store the bytes.
- A flash drive is faster and uses less power than a hard disk. However, per byte, flash is significantly more expensive than hard drive storage. Flash has been getting cheaper. But flash is much slower than RAM so it is not a good replacement for RAM. Flash storage is what underlies USB thumb drives, SD cards for use in cameras, or the built-in storage in a tablet or phone.

File system

The hard drive or flash drive provides persistent storage as a flat area of bytes without much structure. They are formatted with a "file system" which organizes the bytes into the familiar pattern of files and directories, where each file and directory has a somewhat useful name like "resume.txt". Each file in the file system refers to a block of bytes, so the "flower.jpg" name refers to a block of 48 KB of bytes which are the data of that image. The file system gives the user a name (and probably an icon) for a block of data bytes

and allows the user to perform operation on that data, like move it or copy it or open it with a program. The file system also tracks information about the bytes: how many they are, the time they were last modified ...

- Microsoft uses the proprietary **NTFS file system** and MAC OS X has its Apple proprietary **HFS+** equivalent. Many devices (cameras, MP3 players) use the very old Microsoft FAT32 file system on their flash cards. It is old and primitive but it is good where wide support is important.

Bits and Bytes

At the smallest scale in the computer, information is stored as bits and bytes.

Bit

The bit is the smallest unit of storage. It can only store a 0 or 1. Anything with two separate states can store 1 bit. A chip uses electricity 0/1 states. The hard drive uses spots North/South magnetism 0/1 states. A bit is too small to be much use. Better to group bits into bytes.

Byte

One byte = grouping of 8 bits. E.g. 01011010

Number of bits	Distinct Patterns
1	0 1
2	00 01 10 11
3	000 001 010 011 100 101 110 111

⑦ Computer science 101 - Stanford

In general: add 1 bit, double the number of patterns

1 bit = 2 patterns
2 bits = 4 patterns
3 bits = 8
⋮
8 bits = 250

Mathematically: n bits yields 2^n patterns.

One bytes (8 bits) holds a number between 0 and 255

Remember pixel.setRed(n) took a number 0:255 because Red value stored in one byte

Kilobytes Megabytes Gigabytes

The size of information in the computer is measured in kilobytes, megabytes and gigabytes (common sizes)

KB - Kilobyte

1 KB = 1000 bytes.
A small email text is about 2 KB, a 5 page paper might be 100 KB. Text is one of the most naturally compact types of data (1 bytes for one letter).

MB - Megabyte

1 MB = 1 million bytes = 1000 KB
MP3 audio is about 1 megabyte per minute. A high quality digital picture is about 2-5 megabytes.

GB - Gigabyte

1 GB = 1 billion bytes = 1000 MB
Common sized unit modern hardware (2012)

⑧ Computer Science 101 - Stanford

TB - Terabyte

1 TB = 1 trillion bytes = 1000 GB.
It's starting to be common size.

Gigahertz - Speed, not bytes

One gigahertz = 1 billion cycles per second (One megahertz is a million cycles per second).
It is a measure of speed, very roughly the rate that the CPU can do its simplest operation per second. Gigahertz does not precisely tell you how quickly a CPU gets work done but it is highly correlated. Higher gigahertz CPUs also tend to be more expensive to produce and they use more power.

Alternate terminology: Kilibyte Mebibytes Gigibyte Tebibyte

There are two schemes, the "1000" system, and the "1024" system... these result in definitions of M GB TB which differ by up to 10%. In an attempt to fix this, the terms 'kibibyte' "mebibyte" "gigibyte" "tebibyte" have been introduced to specifically mean the 1024 based units.

Software

Software is the general category of code which runs on the hardware. The common case is a "program" like Firefox - software you run on your computer to solve a particular problem. A computer can run multiple

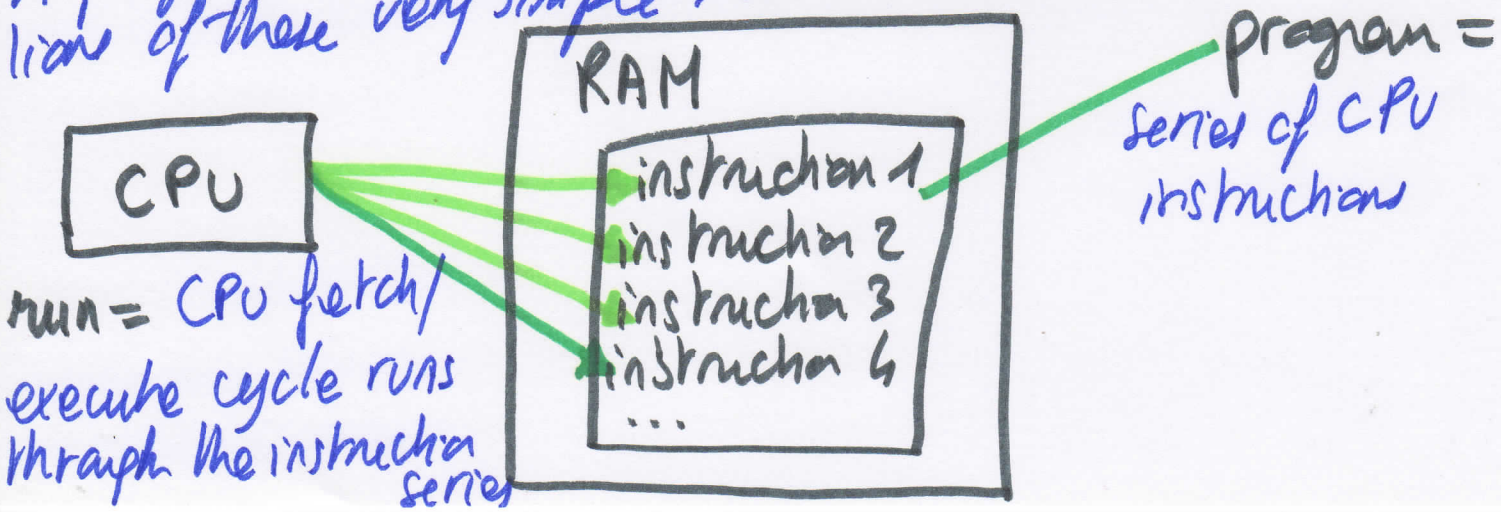
programs at the same time, keeping their use of memory, drawing in windows etc. separated so they hopefully do not interfere with each other.

Machine code

- A CPU understands a low level "machine code" language (also know as "native code"). The language of the machine code is hardwired into the design of the CPU hardware. It is not something that can be changed at will. Each family of compatible CPUs (e.g. the very popular Intel x86 family) has its own, idiosyncratic machine code which is not compatible with the machine code of other CPU families.
- The machine code defines a set of individual instructions. Each instruction is extremely primitive, such as adding 2 numbers. When stored, each instruction takes up just a few bytes. When we say that a CPU can execute 2 billion operations per second, operation refers to these simple machine code instructions

Instructions and programs

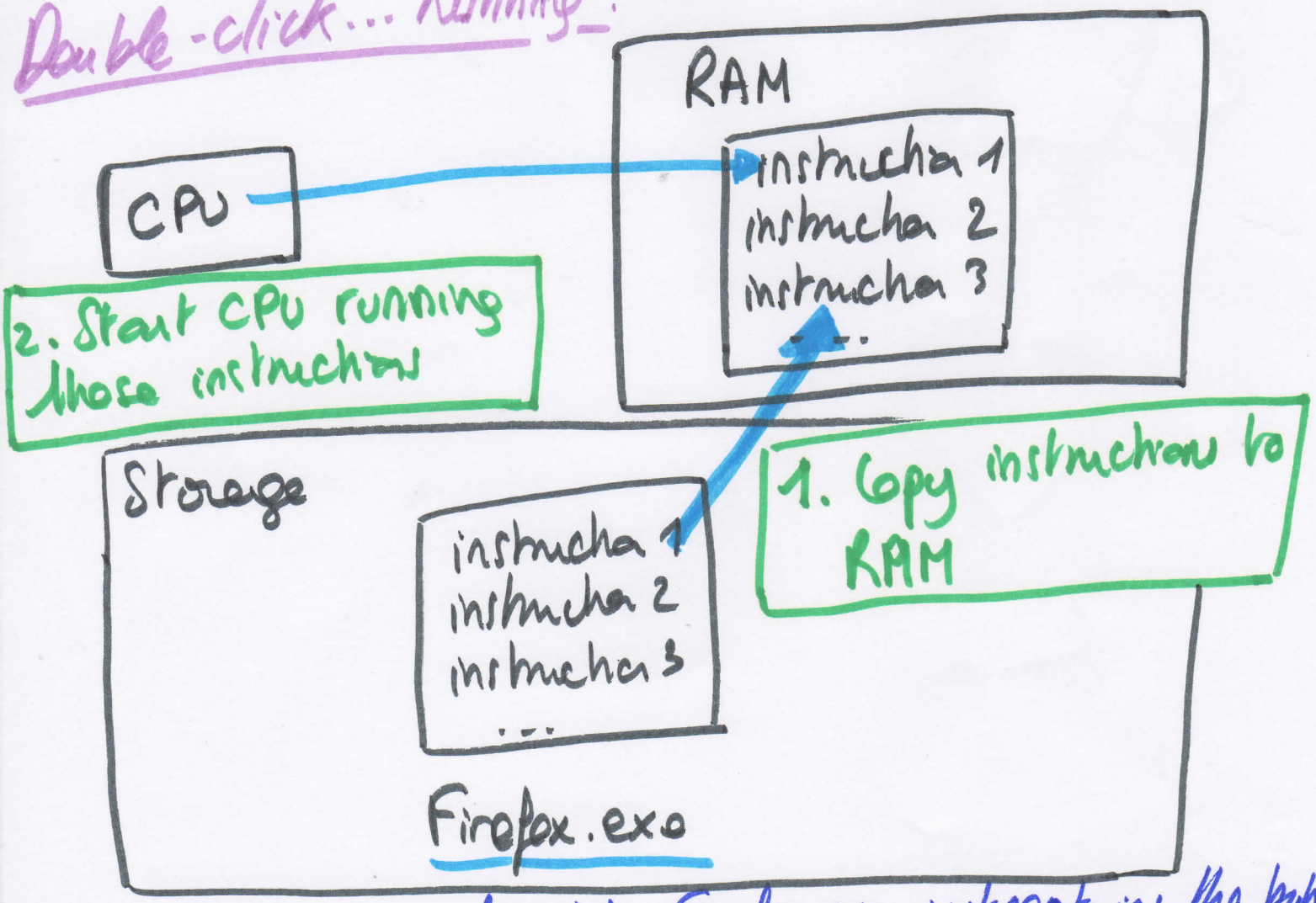
A program, such as Firefox, is made up of a sequence of millions of these very simple machine code instructions.



(10) Computer science 101 - Stanford

The CPU runs instruction using a "fetch-execute" cycle: the CPU gets the first instruction in the sequence, executes it (adding two numbers or whatever), then fetches the next instruction and executes it, and so on. Some of the instruction affect the order that the CPU takes through the instruction sequence: loops direct the CPU to jump back to an earlier point in the sequence and an if statement asks to skip over the next instruction if a particular instruction is true.

Double-click... Running?



In the file system, a file like Firefox.exe just contains the bytes of the machine code instructions that make up the program ("..exe" is a windows convention to mark a file as a program)

(11) Computer Science 101 - Stanford

Each machine code instruction takes up about 4 bytes, and whole program is just an enormous sequence of instructions.
Firefox: about 20 million instructions
When the user double clicks a program file to run it, the block of bytes of the instructions for the program are copied into RAM, and then the CPU is directed to begin running at the first instruction in that area of RAM.

Operating system

- The "operating system" of a computer is like a first, supervisory program that begins running when the computer first starts up ("boots up"). The operating system plays an invisible administrative and bookkeeping role behind the scenes. When a desktop or laptop starts up, the operating system typically gets things organized and then launches a "file explorer" program which displays windows and menus etc. that show the user what file systems are available, allowing the user to navigate and operate on the files.
- The operating system keeps things organized in the background so that multiple programs can run at the same time ("multitasking"). It gives each program its own area of memory, so each program only accesses its own resources.. although to limit what an erroneous or malicious program can do. Each program operates independently, without interfering with other programs or the system as a whole.

(12) Computer science 101 - Stanford

Boot / Reboot

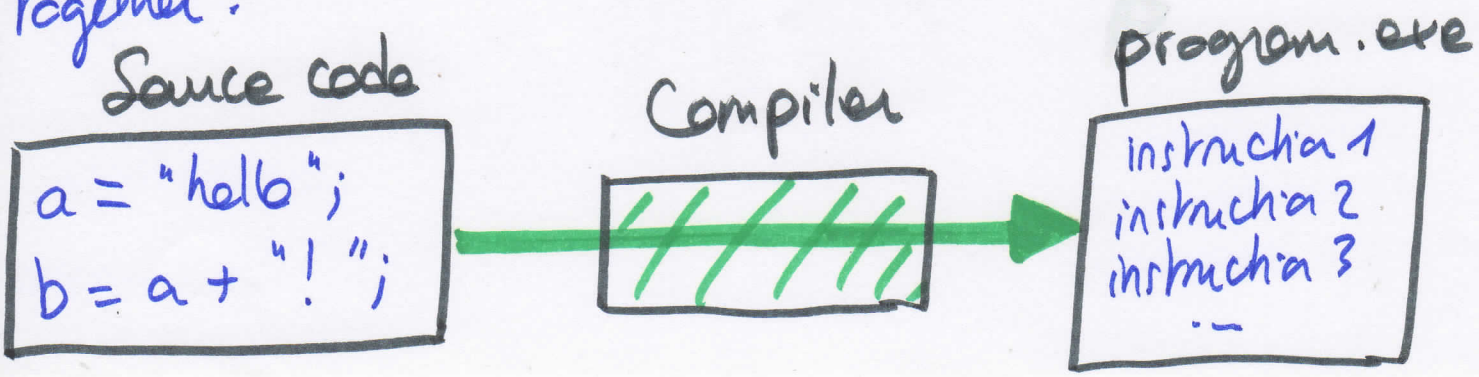
- When first powered on, computer runs a tiny "get started" program that typically looks for a disk containing an operating system to run.
- Bootup = start
- Reboot = shutdown/start fresh cycle

Computer languages

It is extremely rare to write machine code by hand. Instead, a programmer writes code in a more "high level" computer language with features (loops, functions...) that are more useful and powerful. There are two major ways that a computer language can work.

Source code and compiler

One common computer language strategy is based on a "compiler". C and C++ are old and popular computer languages that use this strategy, although they tend to have fewer features than dynamic languages (below). "High level" features can be accomplished by longer sequences of machine code instructions strung together.



(13) Computer science 101 - Stanford

The compiler reads, translates and expands the source code into a longer sequence of the machine code instructions to implement the sequence of actions specified by the source code. (e.g. C++ code). The output of the compiler is, essentially, a program file (.exe or whatever) made of many machine code instructions that implements the actions specified in the source code. The compiler produces the .exe file from the source code and it is finished. Running the .exe can happen later and is a separate step. The compilation step can be done once and is a separate step.

The compilation can be done once and long before the program is run (e.g. produce firefox.exe at Mozilla headquarters)

The end user does not need the source code or the compiler.

The program.exe is in working form.

It does not work backwards, have the .exe, you cannot recover the source code (well).

Source code

The "source code" is the high level code authored by the programmer and fed into the compiler. Generally, just the program.exe file is distributed to users. The programmer retains the source code. Changing the program in the future generally requires access to the source code. For example, to add a feature, the programmer would make changes in the source code, and then run the compiler to produce a new version of the program

(14) Computer science 101 - Stanford

Open source

"Open source" refers to software where the program includes access to its source code, and a license where the user can make their own modification (right to make modif). Typically open source software is distributed for free. It also includes **freedom / independence** since the user is not dependent on the original vendor to make changes or fixes or whatever to the source code. Typically open source licenses include a requirement that such improvements to the source code be made available back to the community at large. Open source is a very successful model for some cases.

Dynamic (interpreter) language

There is a broad category of more modern languages such as Java (world's most popular language), Javascript, and Python, which do not use the compiler/machine code strategy. Instead, these languages can be implemented by an "interpreter".

Interpreter

An interpreter is a program which "runs" other code. For example, web browsers includes a Javascript interpreter. The interpreter runs the source code by taking the lines one at a time, and for each, deconstructs and interprets what the line says to do and then does those actions, piece by piece, in the moment. For example, for "a = 1;", the interpreter reserves a few bytes to store

the value of a, then stores the value 1 into those bytes. Then for "b=a+2;" the interpreter evaluates (a+2) getting the value 3, reserves some bytes for the b variable, then stores the 3 into the b bytes.

Compiler vs. Interpreter

A **compiler** translates all the source code into equivalent machine code program.exe to be run later - it is a **bulk translation**. An interpreter looks at each line of code, and translates and runs it in the moment. The interpreter does not produce a program.exe, instead it performs the actions specified in the source code directly.

Compiled code generally runs faster than interpreted code. This is because many questions - how to append to this string, how many bytes do I need here - are resolved by the compiler at compile time, long before the program runs. The compiler has pre-processed the source code, stripping out many questions and complications, leaving the program.exe as lean and direct as it can be to just run.

In contrast, the **interpreter** deals with each line in the moment, so all the deciphering and overhead costs of interpreting each line are paid as it runs. This makes the interpreted program run more slowly than the equivalent compiled program.

(16) Computer science 101 - Stanford

Dynamic languages - More feature / slower

Dynamic languages tend to have more features (programmer friendly) as they support them more easily.

"**Memory management**" is the problem in a program of knowing, over time, when bytes of RAM are needed and when they can be be reclaimed and use for something else. Every program must solve this problem. Memory management is an excellent example of a feature different between compiled and dynamic languages. Most modern dynamic languages manage memory automatically. The programmer can focus on the problem to be solved, and the dynamic language will take care of managing the memory.

In contrast, in C and C++, the programmer at times must think about memory management and may have to author lines of code to help solve it. (Aside: many crashes in C and C++ programs are due to errors in the programmer's memory management scheme. It is a difficult problem to solve manually).

The memory management in dynamic language is not free. The CPU must run extra lines to solve it. (That's why it's slower)

Dynamic languages allow programmers to produce correct programs more quickly and reliably. As programmers are scarce it is attractive to save some programmer time at the expense of some CPU / memory use (Moore's law reinforced)

Overall, different computer languages have different strengths and weaknesses, and best language for a particular problem depends on the situation.

JIT - Just in Time Compiler

The most modern form of dynamic language is implemented with an interpreter paired with a Just in Time compiler (JIT) trying to get the best of both worlds. The JIT looks a section of dynamic code that are being run very frequently, and for those, does a compile to native code for that section on the fly. So the interpreter is used for simple cases, but for important sections of dynamic code (like the inside of a loop), the JIT creates a block of machine code in RAM for that section. The machine code is run for that section of dynamic code, giving similar performance to C++, and is discarded when the program exits. Java and Javascript both use JIT technology extensively. The great speedup of browsers in the last few years has been largely due to the implemented of JIT technology for Javascript. The JIT erates most but not all of the "10x" penalty. Even with JITs, dynamic languages still have higher overhead use of resources compared to C and C++

(18) Computer science 101 - Stanford

Networking

The computer network (internet) is like a global phone system for computers: a computer can "call" another computer on the internet to get or send a little information. Suppose your laptop is connected to the internet, and you type "http://www.nytimes.com" into your browser - what happens? Your computer contacts the computer "www.nytimes.com" - placing a "call" in effect - and sends a request for the main web page. The request is small, about 1 KB. The www.nytimes.com machine sends back a large response which is the web page - maybe 200 KB - and ends the call. Your browser gets back all this data and formats it for your screen so you can read the text, click links etc. We'll look at this fetch-web-page example a few different ways to see how the internet works.

LAN - Local Area Network

(wire = câble (FR))

Ethernet, wired LAN. Wi-Fi, wireless LAN

Ethernet LAN

Ethernet is an extremely common and influential wired LAN standard. Ethernet cable lengths are typically limited to 100 meters, in keeping with its "local" orientation. A typical LAN application is networking the computers in one room or in one floor of a building. The most common form of ethernet wiring is 100 base-T (100 megabit) with "RJ-45" connectors on the ends. An RJ-45 connector is about the size of your finger.

(19) Computer science 101 - Stanford

Networking / TCP/IP / Analog and Digital

look at the pdf for info

JPEG

Better for real looking photos (texture ...)

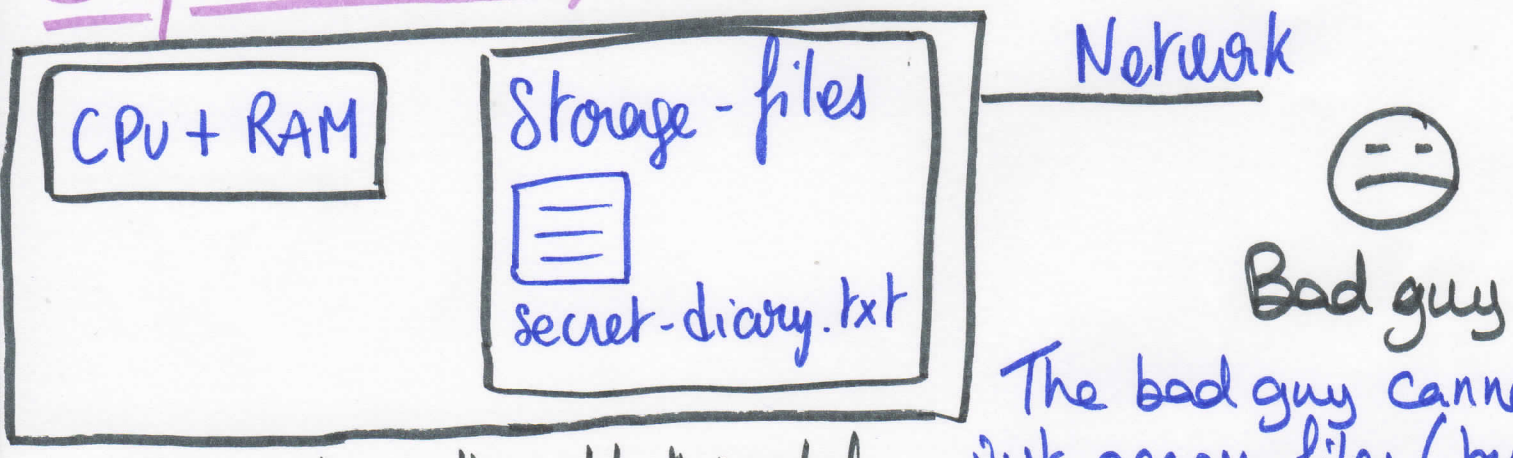
PNG

Better for logos / drawings.. but ok for everything.
Takes less space than JPEG.

Video

2GB/hour.
Video compression is very slow (true for 2012)

Computer security - the castle



Computer - "castle" model

The bad guy cannot just access files (bytes really) stored inside the computer at will. He needs to work at it.

Computer attacks

The three most common type of attack

1. Password attack
2. Phishing attack
3. Malware attack

(20) Computer Science 101 - Stanford

Bad guy attacks

- Typically the bad guys are not crafting some attack just for you. They send in millions rather crude attacks, just snaring the most clumsy victims.
- If you avoid the most common errors, you will probably be fine
- "Spear fishing" refers to sophisticated attacks targeted at a specific person - rare

Password attacks

The bad guy could try to guess you password on a site, trying to log in again and again (every second)

"Dictionary attack": try all the words in a dictionary or/and common used passwords. This fails mostly (99.99%) but success here and there with an account with a poor password.

- See pdf for advice on passwords

Phishing attacks

"Phishing" i.e. the bad guy is "fishing" for you, he tricks you into exposing you password or whatever. A phishing email might say something provocative "fraud alert, click immediately" or points at a "bad guy" site via hyperlink in a fake email (e.g. fake ebay email pointing at a fake ebay website page). When you type you password or so to either of these, it goes to the bad guy.

Note: this is usually not related to an actual transaction or account. The bad guys just spew these out, hoping to fool someone. They mostly target passwords for sites that have something to do with money.

Avoiding phishing

- Don't trust urls or emails or random sites, especially when leading to a login page. (a) Scrutinize the url as shown in the browser (b) (more secure) Type the url in yourself, if it claims to be from ebay, type in www.ebay.com.

HTTPS

HTTPS is the 'secure' variant of HTTP to transfer the bytes of a web page over the internet. It does two things:

1. HTTPS verifies the other end, so it really is www.schwab.com or whatever. Helps with anti-phish, but the user still needs to look at the url. Checking that it's www.schwab.com not www.schwab.bad-guy.ru
2. HTTPS encrypts all the traffic, so interception of the bytes → packet of the communication does not work.

- The server must hold a certificate which is verified beforehand by an authority that the server really represents www.ebay.com or whatever. You have to pay money and file some paperwork to get an https certificate - hard for bad guys to do, although not impossible.

(22) Computer Science 101 - Stanford

écoute illicite

Encryption vs. bad guy packet eavesdropping

When you go on to wifi (Starbucks e.g.) and visit a web page and type something in... the packets for all that are just being broadcast in the room, so anyone nearby can observe the packets, listening in (recall the ethernet packet broadcast). For the most part, this is harmless. In this case HTTPS provides encryption, in addition to its url-verification feature.

Malware attacks

This is a big category, where the bad guy tricks the victim into running bad software ("malware") on the victims computer. We include: viruses, worms, trojans.

Passive content = safe (.TXT .JPG)

Program = unsafe (.DOC .EXE) VBA

"Malware": general term for a program written by the bad guy to do bad things to your machine (break into the machine, steal passwords, send spam, etc.)

- If the bad guy gets you to run bad guy authored code on your computer, the computer is compromized, the bad guy wins. The code can take actions and it's inside the computer. It is like the bad guy is sitting at the keyboard of your unlocked computer, he can do whatever he wants.

Malware 1 - Trojan

A "trojan" is a malware disguised as something else

like "awesome-cursors.exe" or "JustinBieberJPEG.exe". The term refers to the Trojan Horse story from antiquity. So the user downloads it or accepts it, not aware that it will do something bad. The bad guy tries to make it look like harmless content, not a program or they claim to be a program that does something many people want, but it's malware. Operating systems may have helpful warning "This is a program you downloaded, do you want to run it?" Don't run programs from random sources (google it first, see what people say). Someone would flag it if it was malware (if something from a well known domain)

Malware 2 - vulnerability

Suppose there is a **bug** in the Flash animation display program. When fed certain pathologic animation bytes, the program breaks and gives access to the machine. So the bad guy puts a malicious Flash animation, and then sends links to it on sites or in spam. Just visiting the page with the bad content is enough to compromise the machine if it is vulnerable. This is probably the most scary case, as the user does very little.

Solution: Keep web-facing software up to date. All browsers now have strong auto-update channels, so by default the right thing tends to happen. **Aside**: This is why having a proprietary format like Flash be a key part of the Internet

is maybe not a good architecture. People were very dependent on Adobe to fix the software very quickly and Adobe's record was very uneven.
Such vulnerabilities in Flash and IE used to be quite common.